

# Exam Review Class

Programming Languages

# Course

- Programming Paradigms (Declarative Vs Imperative)
  - Chapter 1 -> Foundations of Programming Languages
    - Review Questions are very important
  - You can cross reference with Programming Languages Principles & Paradigms
- Syntax
  - Chapter 2 → Foundations of Programming Languages
    - 2.1, 2.2, 2.3, 2.4, 2.5, 2.11
  - Chapter 2 Programming Languages Principles & Paradigms
    - Page 23 – 37, Page 42-46
- Assembly Language (Dropped from Syllabus)

# Course

- Functional Programming
  - Mathematical Foundations of FPL (PPT shared on GC)
  - Lambda Calculus (Slides 1-2 Part have already shared)
  - Do practice/review questions given on GC
  - Chapter 14 (Programming Languages Principles & Paradigms)
  - Chapter 5 (Foundations of Programming Languages )
- Logic Programming
  - Chapter 7 (Foundations of Programming Languages )
  - Chapter 15 ((Programming Languages Principles & Paradigms)

# Important

- Review Questions of the given chapters are very important
- Read the given chapters thoroughly
- Problem Solving question comprise 60% of the paper
- Definitions / short questions / MCQS/ FB/ TF are also included

# Name the 5 programming domains and languages best suited for each.



- - Scientific (Fortran, ALGOL 60)
- Business (COBOL)
- AI (Lisp, Scheme, Prolog)
- Web (PHP, Java, JavaScript)
- Gaming (C, C++)

# What are the 4 criteria for evaluating programming languages?

- - Readability
  - Writability
  - Reliability
  - Cost
-

# Define orthogonality.

- Small set of primitive constructs can be combined to build the language's data and control structures.

- Prove that the following BNF grammar is ambiguous by showing a string in the language defined by this grammar that has two different parse trees.   Assume that capital symbols are non-terminals and lower-case symbols are terminals and that S represents the start state.

S  $\rightarrow$  aaA

S  $\rightarrow$  aB

A  $\rightarrow$  bB

A  $\rightarrow$  a

B  $\rightarrow$  aA

B  $\rightarrow$  b

# Name that Paradigm

- (a) IMPERATIVE ♦ ♦ What is the name of the category of programming languages whose structure is dictated by the von Neumann computer architecture?
- (b) NON-PROCEDURAL ♦ A paradigm that allows specification of what has to be computed rather than just how a computation is to be carried out.
- (c) OBJECT-ORIENTED A paradigm incorporating encapsulation, inheritance, and dynamic type binding.

# EXAMPLE OF SHORT QUESTIONS

- What are the phases of a compiler?
- What does the syntax analyzer do?
- What does the lexical analyzer do?
- What does the intermediate code generator do?
- What is a purpose of a linker?
- What is pure interpretation?
- What is Java's interpreter called?
- What is a Turing machine?

# Do not Worry!

- Paper is simple
- Only books, books and books....
- 😊

RELAX & STUDY

